

Docket JP920010326US1

Appl. No.: 10/736,343
Filed: December 15, 2003

A computer program iterates through computational loops according to indices, as is well known. A loop may contain "indirect loop index variables," according to the language of claim 1, which refers to variables such as $u(i)$ and $r(i)$ in the following code snippet from Table 1 of the present application:

```
do i = 1, n
  x[u(i)] = . . . .
  . . . .
  . . . .
  y[i] = x[r(i)] . . .
```

See pages 1 and 2 of the original specification.

The method of claim 1 includes "storing . . . a set of unique proxy values" "the unique proxy values being substituted for respective values of indirect loop index variables of the loop" where "each proxy value is a unique prime number." That is, a certain set of unique values are substituted for, and thereby serve as proxies for, the values of indirect loop index variables. See present application, page 10, lines 28-29 ("The use of prime numbers *in place of* the indirect loop index values allows a group of such index values to be represented by a unique number." emphasis added).

The method of claim 1 includes: "calculating . . . indirectly indexed access patterns for respective iterations of the loop" and "the resulting indirectly indexed access patterns have respective . . . pattern values." That is, in a disclosed embodiment, for the collection of all iterations of the loop there is a corresponding set of arrays S_A , S_T and S_F that provide a set of "indirectly indexed access patterns." See present application, page 10, lines 23 -29 (describing arrays S_A , S_T and S_F).¹ For each iteration each one of the three

¹ The elements of S_A include values generated responsive to indirect loop indices for *all* "active array variables," where the loop may include a Boolean condition and the set of all active array variables is the set of array variables defined in assignment statements of the loop body and active either if the Boolean condition evaluates to be true or if the Boolean condition evaluates to be false. See present application, page 9, lines 7 - 31, and equation for array S_A at page 10, line 24. The elements of S_T include values generated responsive to indirect loop indices for *true* "active array variables," i.e., the set of array variables defined in assignment statements of the loop body and active if the Boolean condition evaluates to be true. Id. The elements of S_F include values generated responsive to indirect loop indices for *false* "active array variables," i.e., the set of array variables defined in assignment statements of the loop body and active if the Boolean condition evaluates to be false. Id. Note that S_A may equal S_T or S_F , so that the number of arrays for the indirectly indexed access pattern is effectively two. See present application,

Docket JP920010326US1

Appl. No.: 10/736,343
Filed: December 15, 2003

arrays S_A , S_T and S_F has an element, i.e., a "pattern value" according to the terminology of the claim.

Each indirectly indexed access pattern (each of which has pattern values) is calculated "based upon the unique proxy values for the indirect loop index variables." That is, in the above mentioned embodiment, individual array values are computed for respective iterations of the loop based on the proxy values, which are respectively unique prime numbers, where the proxy values are substituted for the indirect loop indices. See present application, page 10, lines 23 -29 (describing arrays S_A , S_T and S_F). See also, for example, page 18, line 8, through page 22, line 5 (example 1, showing example of substituting prime numbers for indirect loop indices, among other aspects).

Each resulting indirectly indexed access pattern has a given number of pattern values, i.e., "the resulting indirectly indexed access patterns have respective numbers of pattern values" and "none of the respective numbers of pattern values exceeds three regardless of how many statements are in the loop." That is, regardless of the number of statements in the loop body, an indirectly indexed access pattern is a set of a maximum of three arrays, S_A , S_T and S_F according to the embodiment mentioned, where for each iteration each array has an element, i.e., a pattern value, so that for an iteration there may be a maximum of three pattern values. See present application, page 10, lines 23 -29 (describing arrays S_A , S_T and S_F).

Item 2

Examiner Fiegle suggested that Applicant consider adding language to claim 1, for example, stating something about the use of factors of prime numbers. That is, Examiner Fiegle suggested this might make the claim more understandable. While Applicant agrees that the language of the claim is not simple to follow, and while Applicant has seriously considered Examiner Fiegle's suggestion, Applicant is uncertain of specific language that would improve the clarity of claim 1 in this regard, particularly in view of the facts that claim 1 already states that proxy values are prime numbers and that dependent claim 6, which already addresses the

table 14 ($S_A = S_T$). It follows that if the loop includes no Boolean conditions, there is merely one array for the indirectly indexed access pattern.

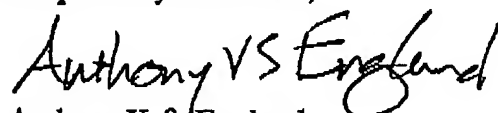
Docket JP920010326US1

Appl. No.: 10/736,343
Filed: December 15, 2003

issue of factors by stating that "the pattern values for respective iterations are calculated by forming *products of the proxy values* of the indirect loop index variables of the loop for the respective iterations."

Nevertheless, Applicant wishes to cooperate fully with Examiner Fiegler and urges him to contact Attorney England at the telephone number indicated below if there is some specific language Examiner Fiegler wishes to propose. Applicant is eager to expedite allowance of the claims.

Respectfully submitted,



Anthony V. S. England
Attorney for IBM Corporation
Registration No. 35,129
512-477-7165
a@aengland.com